

Model Context Protocol (MCP) for AI Systems: A Survey of Architectures and Applications

Neha Upadhyay

Assistant Professor, MANIT Bhopal

ABSTRACT: As Large Language Models (LLMs) continue to advance, the demand for standardized mechanisms that enable seamless, scalable, and context-aware integration of these models in a secure manner is increasing. To meet these requirements, the Model Context Protocol (MCP) has come into existence as an open interoperability standard that offers a common language for linking AI models with tools, resources, and external systems. The survey provides a complete overview of the architectural foundations, communication paradigm, client-server model, transport mechanisms, design principles, and primary application areas of MCP, such as enterprise systems, software engineering, healthcare, manufacturing, education, finance, and multi-agent systems. It also reviews the state of the art and research advances, outlining security issues and their mitigation, and evaluates the status of MCP-based implementations. MCP is shown to be a significant factor in improving interoperability, context-aware reasoning, workflow orchestration, and modular integration of AI, while also minimizing the need for proprietary interfaces. Moreover, it pinpoints key areas of research that are needed in the future, such as scalability, security, standardized benchmarking, interoperability, and long-term context management, offering researchers and practitioners working on the creation of next-generation AI systems significant directions for future research.

KEYWORDS: Model Context Protocol (MCP), Large Language Models (LLMs), Artificial Intelligence, Agentic AI, Context Management, Enterprise Systems.

I. INTRODUCTION

The advent of Artificial Intelligence (AI) and Large Language Models (LLMs) has ushered in a new era of intelligent, interactive systems [1][2]. The Model Context Protocol (MCP) is an open protocol that brings coherence and context to applications. Just like USB-C adds coherence and context to device connectivity, it is important to this in more critical applications including healthcare, finance, legal analysis and real-time decision-making. It's a typical way to connect AI models to external resources and a scalability and interoperability issue because the structure of each external resource is different and the API integration must be different as well [3]. Agent-based workflows are also complicated, and MCP supports them. Through its many pre-built integrations, provider swapping capabilities, and data protection best practices, MCP facilitates structured, stateful interactions among clients, hosts, and servers; these interactions can reveal resources, tools, and prompts that can improve LLMs' inferential capacity [4]. The MCP is an open standard that specifies how LLMs should communicate with other programs, databases, and resources in order to get the right information and control other systems. The MCP standard could be likened to a standard API gateway that enables many tools and data sources to be accessible to assistants. This standardization makes it possible to integrate AI applications with a common pool of MCP servers, enabling common integrations to be reused between different assistants [5]. The host runs an LLM to process user requests and calls on

tools through an MCP client. Access to external systems: Tools to enable external operations are provided by the server. MCP has three core elements: resources (data available), prompts (templates that can be reused for workflow optimization) and tools (functions that can be called).

A. Structure of Paper

The rest of this paper is organized as follows: The Model Context Protocol (MCP) is overviewed in Section II. The MCP Architecture, Communication Framework are discussed in Section III. In Section IV, the key application areas and security issues are discussed. The recent literature is examined in Section V and areas that require research are outlined. Last but not least, Section VI provides a summary of the survey and suggests directions for future research.

II. OVERVIEW OF MODEL CONTEXT PROTOCOL (MCP)

The new interoperability standard called the MCP offers the possibility of better integration of autonomous, reasoning-driven A.I. systems with the corporate software architecture based on deterministic, schema-driven data. Graph endpoints and REST APIs are great for one-step activities, but they can't manage more complicated scenarios like collaborating with other agents, using tools, or retaining states. These conventional user interfaces likewise lack state and are syntactically rigid; they treat each request as a discrete transaction rather than a continuing mental operation. With MCP's semantic, context-aware communication

layer, agents may dynamically negotiate schemas, communicate with other systems in a traceable and policy-compliant manner, and exchange structured context to circumvent this constraint [6]. MCP is essentially a centralized means for encoding and sharing context between diverse tools, application programming interfaces, and agent runtimes. An AI-driven job's development may be described in MCP's first-class context, which includes task intent, tool-generated intermediate reasoning stages, schema definitions, and governance information. The MCP provides a standardized means for storing, transmitting, and retrieving this context, eliminating the need for proprietary memory formats or inconsistent serialization techniques. All interactions with an MCP are related to a persistent Context, thus providing long-lived reasoning continuity between multiple system boundaries.

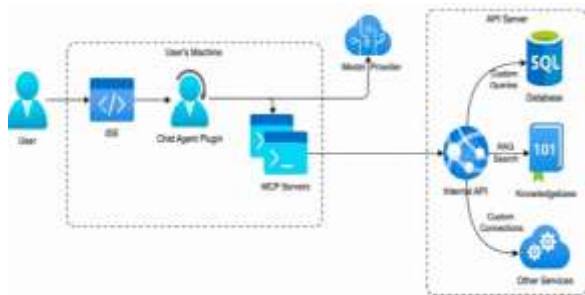


Fig. 1. Model Context Protocol (MCP)

The overall MCP architecture is presented in Fig. 1. depicting the interactions between AI applications, external databases, knowledge bases, and services and how these are connected through MCP servers to facilitate standardized, secure, and context-aware communication between the heterogeneous systems.

A. Design Principles and Objectives

The design of MCP and its architecture and implementation have been guided by the following key design principles:

- **Interoperability:** MCPs can be embedded in different models, platforms and environments, providing consistent management of context across technology implementations. That's helped by the use of common data formats and the protocol's flexibility for languages.
- **Simplicity:** It is designed to be simple, it is easy to implement, it has a limited number of basic primitives that can be built up to create more complex use cases. It helps to lower the obstacles to adoption and promote standardized adoption in various systems.
- **Extensibility:** MCP's extensibility is integrated into its design since its inception. It can be expanded to other areas and additional functionality can be integrated by specified protocols. This stability and evolution make it possible for the protocol to meet new needs and remain compatible.
- **Security and privacy by design:** Security and privacy are not tacked on as afterthoughts, but built-in from the start in MCP. These include data minimization, explicit authorization systems and methods for controlling the information flow between components.

- **Human-centered control:** The protocol ensures that the human is supervised and controlled, especially in sensitive operations or decision-making. This principle does not ignore human judgment in an AI system [7].

III. MCP ARCHITECTURE AND COMMUNICATION FRAMEWORK

A client-server architecture separates the AI models from the resources they use to access data and perform computation. The Model Context Protocol (MCP) is a client-server model, as shown in Fig. 2. In this architecture, a user calls a service from the client and the client calls the MCP server, which calls a variety of external resources and tools. This architectural design has several benefits:

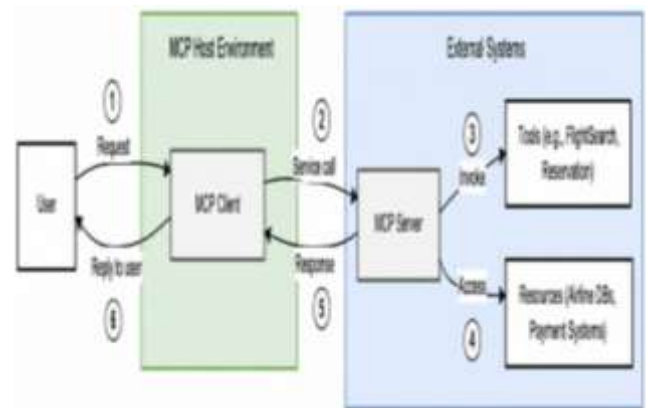


Fig. 2. MCP Client-Server Architecture

- **Clear separation of concerns:** The client-server model separates two concerns: the client takes care of AI reasoning and decisions, while the server takes care of data access and tool execution.
- **Flexible deployment options:** There are several ways to deploy security requirements and operational constraints between clients and servers. They could be on the same computer, on separate networks, or in separate locations.
- **Scalability:** It can be scaled 1-to-many or many-to-one; for example, a single client can connect to numerous servers, or many clients can connect to a single server.
- **Independent evolution:** Keeping them compatible with the protocol, clients and servers can evolve independently, and continue to innovate without stifling the other's progress.

An AI application's client component can connect to one or more MCP servers. This program might function as an assistance interface or an agent framework. All servers are reachable and have access to the same data sources or tools using standard MCP interfaces. The client-server connection is carried out via a lightweight remote procedure call protocol called JSON-RPC. This protocol uses JSON to encapsulate data. This technique provides a universal interface that is independent of languages and may be used in a wide range of contexts.

A. Client Application (Host)

In an MCP environment, the Client Application (Host) is at the very beginning. It maintains the connection to a single or

multiple MCP Servers and controls the flow orchestration according to protocol specifications [8]. The client must create sessions, send and receive resources and tools, prompt or sample, and notify the client asynchronously of events on the server side. Client applications should also provide robust error handling to gracefully degrade when communication fails or times out with the remote MCP Servers if they are meant to interact properly with them.

B. MCP Server

The MCP Server improves LLM processes by supplying data, services, and interaction templates for the client. In addition to showing and controlling contextual resources, it also produces reusable prompts to guarantee consistent interaction instances, uses tools to conduct external activities, and, optionally, lets outsourcing responsibilities related to text production to sampling [9]. The server also processes requests and implements access control policies, keeps track of operational security, and generates notifications to indicate changes in the server's capabilities [10]. This provider-based architecture can be used to modularize access to complex and dynamic resources that complement the client's orchestration logic.

C. Transport Mechanisms (HTTP, SSE, JSON-RPC)

Every client has a unique connection to an MCP server, which can be established using either studio-based JSON-RPC 2.0 (typically used for lightweight synchronous messaging) or Streamable HTTP + Server-Sent Events (SSE), both of which are commonly used for event-driven, asynchronous communication. During setup, components communicate according to the session and protocol rules established by JSON-RPC messaging. The Streamable HTTP transport's robust server-sent interaction and session resumption features rely on unique session IDs and event cursors. The studio transport, on the other hand, allows for fast local execution for edge installations.

D. LLM Integration

LLM invocation layer that intelligently routes sampling requests to either local or remote LLMs based on model preference hints, latency requirements, privacy sensitivity, and cost constraints. Local models include Ollama3, GPT4All4, Llama.cpp5, LM Studio6, and vLLM7. Remote LLMs include those that use enterprise APIs or commercial endpoints. The model identification is not fixed, but rather determined by a preference structure that may consist of factors such as speed, cost, intelligence priority, etc., which allows for it to be used in various scenarios.

E. Security and Modularity

The MCP design is modular and scalable and increases security by separating the context provider, session manager and host. Its extensible protocol design, strict capability negotiation, and flexible transport mechanism allow developers and researchers to create systems with the power of AI, without compromising the privacy of their users. This means that MCP can enable future AI assistants, autonomous agents, IDE-

integrated co-pilots and multi-modal systems to operate with confidence and control in local and distributed environments.

IV. APPLICATIONS OF MCP IN AI SYSTEMS

The uses of MCP in Agentic AI are endless and applicable to a wide range of fields [11]. Developers can easily gain access to relevant code context, repository structure, and product documentation in AI-powered code editors like MCP, which improves developer productivity [12]. In real time, Customer support systems can access customer data, product information and customer support tickets through MCP, and deliver more accurate and contextual Support. MCP's dynamic connection to databases, visualization, and explanation of insights, all while retaining context throughout the analytical process, enhance data analysis workflows.

A. Enterprise Applications

MCP allows AI to search across a variety of document stores, databases, and cloud storage, and tie the results back to the original documents. E-commerce platforms can integrate MCP to analyze customer data in real time and make personalized product suggestions. Smart factories can utilize MCP to connect various machines and systems, allowing for real-time monitoring and optimization of production lines [13].

B. Software Engineering Education

Teaching software engineering through classroom case studies and on-the-job training has long been the backbone of software engineering curriculum reform efforts. A number of recent studies have started using AI technologies like GitHub Copilot to enhance coding productivity and aid in programming instruction [14]. Unfortunately, the protocol cooperation mechanism, which is represented by MCP, is not included in the current educational research, which means that the teaching content is obsolete and the industrial alignment is not excellent [15].

C. Healthcare

In healthcare MCP can facilitate AI-driven diagnostic tools accessing electronic health records, improving the speed and accuracy of medical decisions; it also enables the creation of multi-step, cross-system workflows, allowing AI agents to coordinate tasks across different platforms through a single, standardized interface [16]. Other potential applications are AI agents in smart homes and robotics that understand and interact with their environment, collaborating agents in multi-agent systems, enterprise governance and security systems that standardize access to internal tools with monitoring and control, and personal AI assistants with deep integration into personal data.

D. Multi-Agent Systems

A hybrid design, combining centralized and decentralized techniques of coordination, characterizes most real-world multi-agent systems [17]. Hierarchical in nature, these hybrid models place an emphasis on subgroup coordination at a far lower system level. They can also have a different coordination method for different parts of the system: coordinate strategically at the

central level and coordinate tactically at the decentralized level [18]. Adopting Model Context Protocol sets up standard ways for hierarchical and peer-to-peer relationships to share contexts, which makes hybrid cooperation work better. The result is a greater degree of adaptability in the patterns of coordination, allowing them to change depending on the job and surroundings.

E. Security Challenges and Mitigation Strategies

MCP is a free and open standard that facilitates communication between LLMs and other programs, websites, and databases. This connectivity comes with security challenges in authentication, data privacy, unauthorized access and malicious tool interaction, however [19]. In response, it is important to build AI systems based on the MCP to be secure, reliable and trusted, with its challenges as follows:

- **Unauthorized Access:** Lack of adequate authentication mechanisms can lead to unauthorized access/retrieval of critical resources by users/agents.
- **Data Privacy Risks:** When data is shared between MCP clients, servers, and external services, this could introduce confidential or personal data risks.
- **Prompt Injection Attacks:** Malicious prompts can cause inappropriate behavior in LLMs and may cause unintended tools to be used during execution or information to leak out [20].
- **Tool Misuse:** When tools are compromised or untrusted, harmful actions or manipulation of outputs can occur.
- **Insecure Communication:** Data communication is less secure, allowing data to be more prone to being compromised or tampered with.
- **Identity Spoofing:** An attacker can pretend to be a trusted MCP client/server and access.
- **Third-Party Integration Risks:** MCP ecosystem can be vulnerable due to API/Service integration with third parties.
- **Insufficient Auditability:** There is insufficient logging and monitoring of activities to help detect and investigate security incidents.

F. Mitigation Strategies

To ensure secure interactions with AI, a comprehensive security strategy is required for MCP deployments, consisting of authentication, encryption, monitoring, and policy enforcement. Strategies include:

- **Strong Authentication:** Authenticate users and services with OAuth 2.0, API keys and multi-factor authentication.
- **Role-Based Access Control (RBAC):** Users are granted access to resources by roles and permissions.
- **End-to-End Encryption:** TLS/SSL encryption of communication for secure data transmission.
- **Prompt Validation and Filtering:** Identify and prevent harmful or threatening prompts from being executed.
- **Tool Verification:** Tool Verification (Verify tools prior to calling into operation, to ensure they are identified, have proper permissions, and are not corrupted).

- **Secure API Gateway:** Authenticate, authorize and rate limit API calls, and monitor it all from one place.
- **Comprehensive Logging and Auditing:** Keep holistic logs to monitor, comply and investigate forensically.
- **Continuous Security Monitoring:** Set up anomaly detection to identify any suspicious activity in real time.
- **Regular Security Updates:** Patch vulnerabilities and update MCP components, APIs and dependencies regularly for system resilience.

V. LITERATURE REVIEW

In this section, the authors review the latest studies on the MCP for AI systems, highlighting main architectures, applications, and gaps. The reviewed studies and their methodologies, findings, advantages, challenges and recommendations are compiled in Table I.

Fernandes et al.(2026) propose an LLM-powered chatbot that enables stakeholders to pose questions to the supply chain in natural language. The structured reasoning is handled by LangGraph, while the live voice in several languages is handled by Gemini Live API. Documents, APIs, and blockchain event logs can all be integrated into the system using MPC, and information may be protected by role-based access restrictions. A concept is proposed that demonstrates that users who are not familiar with blockchain can gain access to verifiable information about a supply chain through a natural dialogue. The PoC demonstrates that blockchain-accessible verifiable supply chain information can be delivered through natural dialogue without blockchain knowledge, which could serve as a basis for an easier decentralized traceability system [21].

Farook et al. (2026) provide an overview of the MCP, detailing its features, operation, and server lifespan. Information retrieval, question responding, and picture analysis are all made possible by the system's usage of MCP and the Agent-to-Agent (A2A) protocol, which allow for natural language communication and multimodal answers. In terms of BERT Score F1 (95.4%) and LLM-as-a-Judge Eval (87.6%), the experimental findings demonstrate high levels of inter-agent synchronization, task allocation, and domain-specific response production [22].

Gao et al. (2025) unveil a conversational intelligence architecture utilizing the MCP and MCPO. The MCP tools are encased as RESTful APIs that comply with OpenAPI standards, and a proxy server from MCPO is utilized to facilitate communication. The architecture promotes interoperability, security, scalability and reduces deployment dependency so that tools are easily integrated. The experimental findings show that it is beneficial for creating conversational intelligence systems that are safe, dependable, and scalable [23].

Vieira Da Silva et al. (2025) suggest using an MCP-based approach that allows the LLM agents to interact with manufacturing resources via a uniform interface. The prototype evaluation is conducted on a lab-scale manufacturing system, where effective multi-step planning, handling of manufacturing constraints, and invoking manufacturing resources are

demonstrated by using MCP. The findings emphasize the feasibility of using MCP for flexible industrial automation systems, without the necessity of having explicit semantic models, paving the way for future advancement of LLM-based production systems [24].

Jadav (2025) The design incorporates MCP Servers, Hosts, and Clients via an API Gateway that utilizes TLS 1.2, OAuth 2.0, and enterprise authentication. User intent, tool selection, and the LLM-generated reply controlled by the MCP Host, and an MCP Inspector keeps an eye on the behavior of the tools. Experimental results show secure, scalable and compliant AI-

agent workflows, dynamic tool discovery, secure execution and minimized unauthorized actions and data leakage [25].

Altinors and Akgun (2025) introduce MemCP, a server capable of serving as a production server that provides permanent memory for AI assistants that support the MCP. The seamless integration of AI models and external services is made possible by its dual-server design, hybrid storage, and asynchronous processing. The results of the experiments show that, despite the limitations of the tool calling and token dependency aspects of AI use, standardized AI context management is possible and useful. [26].

TABLE I. SUMMARYRYSUMMERY OF THE STUDY ON MODEL CONTEXT PROTOCOL (MCP) FOR AI SYSTEMS

Authors	Approach	Findings	Advantages	Challenges	Recommendations
Fernandes et al. (2026)	LLM chatbot with MCP, LangGraph, Gemini Live API	Natural language access to blockchain-based supply chain data	Improves accessibility, multilingual interaction, secure traceability	Limited evaluation in large-scale real-world supply chains	Validate in industrial environments with diverse stakeholders
Farook et al. (2026)	MCP integrated with A2A protocol for multi-agent communication	Effective inter-agent coordination and multimodal task execution	High interoperability and response quality	Limited domain-specific validation and scalability analysis	Extend evaluation across heterogeneous AI ecosystems
Gao et al. (2025)	MCP tools exposed through MCPO and OpenAPI REST APIs	Enhanced interoperability, security, and scalable tool integration	Simplifies deployment and tool management	Performance under complex distributed environments not explored	Optimize large-scale deployment and runtime efficiency
Vieira Da Silva et al. (2025)	MCP-enabled LLM agents for manufacturing automation	Supports planning and resource invocation without semantic models	Flexible industrial automation and standardized resource access	Prototype limited to laboratory-scale manufacturing	Evaluate in real industrial production environment
Jadav (2025)	Secure MCP architecture with API Gateway, OAuth, and Inspector	Secure and compliant AI-agent workflows	Strong security, authentication, and governance	Increased architectural complexity and deployment overhead	Improve lightweight security frameworks and automation
Altinors & Akgun (2025)	MemCP persistent memory server for MCP assistants	Standardized long-term context management	Persistent memory and seamless service integration	Higher token usage and dependency on tool calls	Develop efficient memory optimization and context compression

Research Gap: MCP has shown great promise in integrating LLMs with external tools and systems, but there are still some issues to be addressed. Much existing work focuses on prototype systems, and large-scale deployment and performance evaluation is relatively limited. The use of prompt injection attacks, unauthorized access to tools and privacy issues remain important security challenges. Furthermore, there are insufficient solutions for interoperability among various implementations of the MCP, efficient context management is not yet fully solved and scalable multi-agent coordination is not yet fully addressed. A standard structure, more secure structures must be created to allow widespread use of domain-specific MCP applications and additional research must be performed.

VI. CONCLUSION AND FUTURE WORK

The state-of-the-art development of an AI that is able to reason, make decisions and autonomously perform tasks in different application areas. For scalable, secure, and context-aware integration of AI to rely on external tools, data sources, and enterprise services, standardized interoperability is essential. This survey thoroughly examined the MCP including its architectural foundations, communication framework, client-server architecture, transport mechanisms, and major application areas. It also covered the latest research advances and how MCP is advancing interoperability, workflow orchestration, multi-agent collaboration and context management, and seamless integration with heterogeneous systems. The survey also

pinpointed the greatest security issues and solutions. To summarise, the status quo of MCP provides a good foundation for next-generation AI systems, but issues such as scalability, interoperability, security, efficient context management and others may prove difficult for widespread use in industry.

Future real-world applications of such systems will be seen as scalable, performance metrics will be acceptable and have a clear meaning, multi-agent coordination will be provided, context will be managed efficiently, and communication will be guaranteed with privacy. Reliable and trustworthy AI ecosystems will be widely adopted as there is an increasing number of explainable AI systems and everyone will be able to use it on any platform, it will be easier to create security structures around it and deploy specialized MCP.

REFERENCES

1. G. Qu, Q. Chen, W. Wei, Z. Lin, X. Chen, and K. Huang, “Mobile Edge Intelligence for Large Language Models: A Contemporary Survey,” *IEEE Commun. Surv. Tutorials*, vol. 27, no. 6, pp. 3820–3860, Dec. 2025, doi: 10.1109/COMST.2025.3527641.
2. A. Singh, A. Shetty, A. Ehtesham, S. Kumar, and T. T. Khoei, “A Survey of Large Language Model-Based Generative AI for Text-to-SQL: Benchmarks, Applications, Use Cases, and Challenges,” in *2025 IEEE 15th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, Jan. 2025, pp. 00015–00021. doi: 10.1109/CCWC62904.2025.10903689.
3. K. R. Kiran, “API Integration Barriers in Case-Based Process Management: A Review of Interoperability and Real-Time Response Constraints Outline,” *Zenodo*, vol. 25, no. 03, 2025, doi: 10.5281/zenodo.16782616.
4. P. P. Ray, “A Survey on Model Context Protocol: Architecture, State-of-the-art, Challenges and Future Directions,” Apr. 18, 2025. doi: 10.36227/techrxiv.174495492.22752319/v1.
5. H. Errico, J. Ngiam, and S. Sojan, “Securing the Model Context Protocol (MCP): Risks, Controls, and Governance,” 2025, doi: 10.48550/arXiv.2511.20920.
6. P. Venkateela, “The New Interoperability Paradigm: Model Context Protocol (MCP), APIs, and the Future of Agentic AI,” *Comput. Fraud Secur.*, pp. 1259–1271, May 2025, doi: 10.52710/cfs.817.
7. V. S. Narajala and I. Habler, “Enterprise-Grade Security for the Model Context Protocol (MCP): Frameworks and Mitigation Strategies,” in *2026 IEEE 5th International Conference on AI in Cybersecurity (ICAIC)*, IEEE, Feb. 2026, pp. 1–8. doi: 10.1109/ICAIC67076.2026.11395723.
8. S. Singh, S. A. Pahune, P. Chatterjee, and R. Sura, “Advanced Machine Learning Techniques for Prediction of Customer Churn in Telecommunication Sector,” in *2025 IEEE 6th India Council International Subsections Conference (INDISCON)*, Rourkela, India: IEEE, 2025, pp. 1–6, August. doi: 10.1109/INDISCON66021.2025.11252233.
9. P. Chauhan, U. P. Rao, and M. M. Kailashchandra, “MCP-PUF: A Context-Hardened, ML-Assisted Authentication Protocol for Secure Healthcare IoT,” in *2025 IEEE 4th International Conference on Technology, Engineering, Management for Societal Impact using Marketing, Entrepreneurship and Talent (TEMSMET)*, IEEE, Oct. 2025, pp. 1–6. doi: 10.1109/TEMSMET65536.2025.11467437.
10. A. Ehtesham, A. Singh, G. K. Gupta, and S. Kumar, “A survey of agent interoperability protocols: Model Context Protocol (MCP), Agent Communication Protocol (ACP), Agent-to-Agent Protocol (A2A), and Agent Network Protocol (ANP),” pp. 1–21, 2025, doi: 10.48550/arXiv.2505.02279 Focus to learn more.
11. K. Jangiti, “Design and Validation of a Machine Identity Governance Framework for AI Agents in Multi-Cloud Environments,” in *SoutheastCon 2026*, Huntsville, AL, USA: IEEE, 2026, pp. 1–6, February-March. doi: 10.1109/SoutheastCon63549.2026.11476363.
12. G. Tod-Raileanu, S.-D. Axinte, A.-M. Dincă, and I. C. Bacivarov, “Security Posture Evaluation of Model Context Protocol Servers Against Prompt Injection,” in *2025 IEEE 31st International Symposium for Design and Technology in Electronic Packaging (SIITME)*, IEEE, Oct. 2025, pp. 249–252. doi: 10.1109/SIITME67657.2025.11293682.
13. V. Ayyagari, “Model Context Protocol for Agentic AI: Enabling Contextual Interoperability Across Systems,” *Int. J. Comput. Exp. Sci. Eng.*, vol. 11, no. 3, pp. 6072–6082, Aug. 2025, doi: 10.22399/ijcesen.3678.
14. A. Shah, A. Chernova, E. Tomson, L. Porter, W. G. Griswold, and A. G. Soosai Raj, “Students’ Use of GitHub Copilot for Working with Large Code Bases,” in *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*, New York, NY, USA: ACM, Feb. 2025, pp. 1050–1056. doi: 10.1145/3641554.3701800.
15. K. Chen, J. Keung, W. Li, X. Shao, Y. Li, and X. Ma, “Teaching Software Engineering with LLM and MCP Integration: From Classroom to Industry Practice,” no. 3, 2026, doi: 10.48550/arXiv.2606.19167.
16. M. Mittal, “AI in Healthcare: The Next Frontier In Medical Diagnosis,” *Int. J. Inf. Technol. Manag. Inf. Syst.*, vol. 16, no. 2, pp. 1448–1455, Mar. 2025, doi: 10.34218/IJITMIS_16_02_091.
17. M. R. C. Mukkolakkal, “Deploy, Calibrate, Monitor, Heal -- No Human Required: An Autonomous AI SRE Agent for Elasticsearch,” 2026.

- doi: 10.48550/arXiv.2604.03933.
18. N. Krishnan, “Advancing Multi-Agent Systems Through Model Context Protocol: Architecture, Implementation, and Applications,” *arXiv Prepr. arXiv2504.21030*, Apr. 2025, doi: 10.48550/arXiv.2504.21030.
19. Z. Zhou *et al.*, “MCPShield: A Security Cognition Layer for Adaptive Trust Calibration in Model Context Protocol Agents,” 2026, doi: 10.48550/arXiv.2602.14281.
20. S. P. Sivashanmugam, S. Kumara, A. Mohile, and D. R. Suram, “Improving Detection Accuracy of Phishing Attacks with Feature Selection and Machine Learning Techniques in Cybersecurity,” in *2026 1st International Conference on Emerging Trends in Advancements and Applications of Computational Intelligence Techniques (ETAAct)*, Bhubaneswar, India: IEEE, Apr. 2026, pp. 1–6. doi: 10.1109/ETAAct69135.2026.11542138.
21. A. Fernandes *et al.*, “Evaluating Model Context Protocol in the context of a Conversational Agent supporting an Agro-Food Blockchain Logistics Platform,” *Procedia Comput. Sci.*, vol. 278, pp. 742–750, 2026, doi: 10.1016/j.procs.2026.03.046.
22. M. S. Farooq, M. Sarwar, N. A. Nawaz, U. Ahmed, M. H. Jamil, and J. Ahmad, “Advanced LLM-based Model-Context-Protocol and Agent-to-Agent Protocol for Child Medical Emergency,” in *2025 6th International Conference on Innovative Computing (ICIC)*, IEEE, Dec. 2025, pp. 1–6. doi: 10.1109/ICIC68258.2025.11413209.
23. M. Gao, L. Li, X. Ming, and G. Shen, “Research on Standardized Encapsulation of Model Context Protocol Tools and AI Agent Integration,” in *2025 International Conference on Aerospace Information Perception and Intelligent Processing (AIPIP)*, IEEE, Oct. 2025, pp. 328–331. doi: 10.1109/AIPIP66876.2025.11299220.
24. L. M. V. da Silva, A. Köcher, and F. Gehlhoff, “Beyond Formal Semantics for Capabilities and Skills: Model Context Protocol in Manufacturing,” in *2025 IEEE 30th International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, Sep. 2025, pp. 1–4. doi: 10.1109/ETFA65518.2025.11205601.
25. V. Jadav, “Multi-Context Protocol (MCP) Hosts provide secure and real-time integration of AI tools,” in *2025 International Conference on Computer and Applications (ICCA)*, IEEE, Dec. 2025, pp. 1–6. doi: 10.1109/ICCA66035.2025.11430968.
26. A. Altınörs and B. T. Akgün, “A Model Context Protocol Application for Long-Term Memory Management,” in *2025 3rd International Conference on Foundation and Large Language Models (FLLM)*, IEEE, Nov. 2025, pp. 790–797. doi: 10.1109/FLLM67465.2025.11391050.